

**The Redesign and Reimplementation of the Stevens Institute of Technology Campus  
Network**

*Group 12  
May 2, 2003*

**Faculty Advisor**  
Prof. Sumit Ghosh

**Group Members**  
*Grace Shabo  
Nicholas Evans  
Philip Tan  
Jay Grossman  
Hans Kim*

We pledge our honor that we have abided by the Stevens Honor System.

## Table of Contents

I.	Abstract	1
	1. Acknowledgement	2
II.	Implemented Prototype	
	1. Introduction	3
	2. Prototype Specification	5
	3. Prototype Performance and Evaluation	8
	4. Financial Budget	12
	5. Project Schedule	13
III.	Conclusion	14
IV.	References	16
V.	Appendices	
	A. FreeBSD Scalability Options	
	B. FreeBSD Usability and Ports	
	C. thttpd	
	D. Postfix	
	E. FreeBSD Kernel Configuration Files	
	F. Postfix Configuration File (Main.cf)	
	G. Performance Index	

## **I. Abstract**

We present the implementation of our design for the next generation campus network for the Stevens Institute of Technology. Due to the large scope of such an undertaking, the concentration of the project was the rebuilding of the main campus server known as Attila. Attila, currently a single SGI Challenge computer, was redesigned as a set of distributed servers in order to increase performance, robustness, and reliability. Load balancing was implemented in order to further increase the system's throughput. Our results were good. We achieved better than expected results when testing SMTP and HTTP servers. Our SMTP results equate to every user on campus (~6700) being able to send approximately 35 e-mails per minute continuously. HTTP results were excellent at 55-60 megabits of sustained data transfers per second equaling approximately 200,000-300,000 hits per hour. Our POP3 results were mixed, yet still promising considering the test systems are only 350 MHz Pentium II systems.

## **1. Acknowledgement**

The group would like to thank the Electrical and Computer Engineering (ECE) Department for providing the funds necessary for constructing some of the servers required for the project. We would also like to thank Professor Ghosh for his advice throughout the course of this project. We would like to thank Chris Kelly of Veracast Communications, Inc. for lending us the Cisco 2924XL switch used during our simulation.

## **II. Implemented Prototype**

### **1. Introduction**

As described in the final report for the fall semester, Attila is severely overloaded. Services such as e-mail, terminal logins, various campus web sites (academic, club, and personal student web sites), and file storage space all reside on Attila. This overloading was demonstrated through the use of Attila's relatively slow responses to hourly pings by a machine set up in Technology Hall. In addition, a catastrophic failure of Attila's hardware or software would cause chaos on the Stevens campus.

Web-based courses are being pushed heavily at Stevens, so a natural extension of these courses would be on-demand streaming videos of lectures. Currently the networking infrastructure does not have the capacity to handle multiple video streams with decent quality for all simultaneous users. This limitation was addressed in our prototype as well.

The next generation campus network utilized a load balanced approach to service availability. Since Attila lacks redundancy and scalability, our prototype was built with these key points in mind. A load balanced, distributed web serving setup was created to allow for higher performance and scalability. Also, having this type of setup allows for a built-in failsafe should one of the systems go out of service. Our results demonstrated that a very high level of performance can be obtained from relatively low-powered hardware while also providing the reliability not present in the current implementation. Some of the same concepts implemented and tested in our design are used by many successful commercial entities such as Yahoo and Google. The idea of load balancing inexpensive hardware is proven to be a viable alternative to big-iron hardware similar to the SGI system that constitutes Attila now.

To accommodate the campus's needs for streaming video, prototype RealNetworks RealServer streaming media servers were tested to demonstrate the ease and feasibility of implementing such a system. Our tests were conducted with realistic video streams which approximated the length of many on-campus lectures. System performance was better than expected, with relatively high demands placed on the servers.

Our mail server testing provided some striking results. We were able to service a large amount of e-mail traffic using the Postfix mail transfer agent (MTA). We were able to serve enough traffic such that each user on the Stevens network could send a sustained 30-40 e-mails per minute. This was done on our modest hardware, and with some subtle changes to that hardware, we could have served even more. This represents one of the tenets of our design: to use quality open source software configured such that it can serve large amounts of users in a stable, speedy fashion.

Currently Attila provides the bulk of the campus's network-based services. Major services such as e-mail, personal web space, shell accounts, FTP access, and many club and infrastructure web services (Registrar's website, for example) are served from Attila. We tested load balanced alternatives for SMTP for mail transfers, POP3 for e-mail downloads, HTTP for both student websites as well as infrastructure sites, RealServer for streaming video, and, through some of our benchmarking tests, conclusions can be drawn about shell account performance. Overall our test results were better than expected. We feel for some services a minimal set of hardware requirements would be sufficient to serve the needs of the Stevens community while others may require stronger systems or a greater quantity of systems to serve content at the speed desired.

## 2. Prototype Specification

The prototype server cluster to replace Attila consists of several computers. Group members own some of the machines, while others were built using spare parts and parts purchased using the senior design stipend allowed to each group. The two main servers were Intel Pentium II 350 MHz computers outfitted with identical SCSI cards and hard drives. Identical systems were used for the load balanced group so that results could be easily scaled based on our performance index. A load-balancing server was built to do round-robin load balancing for the web hosting servers. Five load applying systems were constructed to send HTTP requests to the web servers. All of the machines were connected together with a Cisco 2924XL 24-Port 10/100 Fast Ethernet switch. All systems were configured with the FreeBSD 4.7 operating system.

Each load balanced group server was configured with similar hardware. We used two Pentium II 350MHz systems each with 256 Megabytes of memory. Both units had identical disk subsystems consisting of Adaptec 2940UW PCI Ultra-Wide SCSI controllers mated to Compaq/Fujitsu 9 Gigabyte, 10,000 RPM Ultra2 SCSI drives. Both systems were configured to run at 350MHz with a 100MHz front-side-bus. Memory performance was tuned through BIOS options to the highest performance settings allowed by the motherboard.

The "load balancer" consisted of a Celeron 533 MHz processor with 256 Megabytes of memory. Again the best performance options available in the BIOS were used, of course providing they did not impact system stability. This system was configured with FreeBSD 4.7 as well as IPFilter 3.4.29. The load balancing scheme used included a round-robin approach using IPFilter's built in Network Address Translation (NAT) system. While this subsystem was designed to do NAT and port redirection, the author saw the opportunity to implement

rudimentary load balancing features. The external IP Address on the load-balancer was on an Internet Assigned Numbers Authority (IANA) reserved network, 192.168.0.0/24 with an address of 192.168.0.242 (known as GLB). Our test network was on another IANA reserved network of 10.0.0.0/24 with our first load balanced cluster server being 10.0.0.10 (known as SERVER1) and our second server being 10.0.0.11 (known as SERVER2). Our internal load balancer interface was their gateway to the adjoining networks, addressed at 10.0.0.1 (known as GLB-INT). Configuration of our load balancer was a simple matter of adding a line to our IPNat configuration:

**rdr fxp0 192.168.0.242/32 port 80 -> 10.0.0.10,10.0.0.11 port 80 tcp round-robin**

The line above is broken down as follows. "rdr" directs the filter to redirect the packets described in this line. "fxp0" is the system name for the network interface; in this case it is an Intel 10/100 Server Adapter using the fxp driver in FreeBSD. "192.168.0.242/32" is the address on which translations were performed. "port 80" is the port on which any translation will be performed. "-> 10.0.0.10,10.0.0.11" indicates that we are to direct incoming traffic on GLB (192.168.0.242) port 80 to the IP addresses for SERVER1 and SERVER2. "port 80" directs the filter to send translated packets to port 80 on the destination address(es). "tcp" specifies the protocol to translate; in this case, UDP ports sent to port 80 would not be translated. "round-robin" specifies that the filter is to use the round-robin delivery scheme on the list of IP addresses given. Round-robin delivery will deliver the first session to the first server, the second session to the second server, then back to the first server for the third session and so on. This is essentially a "poor-man's" approach to load balancing. The unit recommended in previous design reports is the F5 BigIP load balancer (<http://www.f5networks.com/f5products/bigip/>), which is an

industrial strength load balancer, offering a wide range of configuration and balancing options. Since such a unit could not be acquired without considerable financial resources we were forced to come up with our own solution. For the most part the solution was successful, but problems did arise with certain tests that the F5 unit does not have trouble with.

To test the mail server configuration, a separate machine was built utilizing both spare parts and purchased parts needed to complete the machine. This machine was an AMD Athlon 750 MHz outfitted with 128 MB RAM and a 7 GB hard drive. It was also configured with FreeBSD 4.7. The Postfix mail suite was used for this prototype. It was configured to accept mail for a fictitious domain "seniord.net". After configuring for this option a virtual list of users was created by using Attila's /etc/passwd file to gather usernames. 5% of the ~6700 users on Attila were simulated, which equaled approximately 350 users on the test servers. Since the /home directory was created on an NFS file server, all that was done was to create the userlist and add the users on the server. This was done on SERVER1, followed by copying password files and any other user relevant information to SERVER2. This created identical user logins on each server. /home was mounted on SERVER1 and SERVER2 from the external file server via NFS. Each system was able to operate and access user data. Although only one server was used in testing to simplify configuration, these could have been run in a load balanced fashion. See Appendix D for specific mail server configuration settings.

On a general note, all FreeBSD systems were configured according to the needs of the server. The SERVER1 and SERVER2 systems had a base system installed along with several support software packages. Custom system kernels were compiled that included only the essential hardware. Unused drivers and subsystems were removed to save memory and increase

base system performance. Since all systems were of Intel i686 architecture and above, lower architecture support in the kernel was removed by commenting out:

```
#cpu      I386_CPU  
#cpu      I486_CPU  
#cpu      I586_CPU
```

in the kernel configuration files. Complete kernel configuration files are included in Appendix F. All system compiler options were left at their default values. As discussed in Appendix A there is potential for an additional performance boost by adjusting compiler options to generate faster byte-code. There is also the potential for a sizable increase in performance by simply recompiling the base FreeBSD system with these optimized compiler flags.

### **3. Prototype Performance and Evaluation**

#### **3.1 thttpd**

thttpd testing was performed in the load balanced group using Jef Poskanzer's http\_load program. http\_load is a flexible program that performs HTTP requests and can do operations on the requested files to verify their safe arrival. http\_load requires a few command options including a URL list which it will then attempt to fetch. http\_load is a free application and is capable of generating a significant amount of HTTP traffic even on modest hardware. To provide content for the tests performed, several pages of the Stevens website were mirrored and placed in the content root directories on each web server. Mirroring was done with the GNU Wget program, which will rewrite files such that they use relative links so local browsing does not require Internet access. All files can be viewed locally without the need to connect to a remote server to access a link. After placing these files in the content directories of the web servers, a list

of URL's was built to feed into http\_load. These URL's used the GLB IP address of 192.168.0.242 so that IPFilter would perform round-robin load balancing.

The results were excellent, but not without problems. Initial tests achieved approximately 90 megabits of continuous HTTP traffic from the 2 load balanced servers (SERVER1 and SERVER2). Each server at this point was only using about 40-50% CPU time with load averages around 1-2 points. This level of performance was astonishing, and while it was partially expected, it was not long lived. After approximately 20 minutes of sustained load, the servers became inaccessible from the network. Originally, the switch was suspect, but after further investigation it was discovered that the memory resources of the servers were being exhausted. After waiting approximately 10 minutes the servers were accessible again. This was curious to us, but we persisted. Some load clients were removed, and the tests were run again. This time approximately 70 megabits of traffic was simulated for about 25 minutes. Both servers froze again, and after waiting another 10 minutes, they returned to their normal state. Logs were checked, and it was found that the systems did not actually freeze up. The local system logs stated that MBUFS had been exhausted. This immediately indicated that the network buffers were all used up, as discussed in Appendix A. This was verified by running "netstat -m" and "vmstat -m" on FreeBSD which gives statistics about network and system memory usage. The log findings made more sense since local logins to the system could be performed, but not network logins. The solution was to increase the number of buffers allocated to the kernel for networking. Appendix A outlines the steps performed. The network buffers were increased, and more memory was allocated to the kernel. Testing continued following these modifications.

The overall results were more subdued because while kernel memory and buffer space were increased, it was not long before they were exhausted again. 55-60 Megabits of HTTP

traffic were simulated without each server freezing. CPU usage at this rate was approximately 10-20% on SERVER1 and SERVER2 with load averages around 1 point. The results would have been greater if each server had more memory (512 MB-1 GB); however, the results achieved are excellent for Stevens campus use. Since off-site connections would be limited to the 15 Megabit Verizon DS3 link, the remainder (40-45 Megabits) would be available for on-site use. This easily exceeds any current needs of the campus network. To note, these results are also achieved with 2 Pentium II 350MHz systems. Modern server hardware can easily outperform these units.

### **3.2 Postfix**

The Postfix testing went smoothly however the configuration created some problems. Configuration proceeded according to several manuals and user experiences on the Internet, but the server refused to receive mail. Postfix was configured and listening on TCP port 25 as it should be, but it would not service requests. Telnet-ing to port 25 on SERVER1 was attempted to manually feed Postfix SMTP commands, but no responses were received. After searching through mailing list archives it was discovered that, since a mail alias database was not created from the default aliases file, Postfix would refuse to acknowledge any SMTP commands. After correcting the problem Postfix answered requests and load testing began.

A program called “postal” was used to perform load testing. It supports SMTP and POP3 protocols for sending and receiving e-mail. The same list generated for our user creation was used to create an e-mail address list that postal would send mail to. It generates messages of random size and sends them to users at random from the e-mail address list. Several test “mixes” were run in which the number of messages sent per connection and the number of connections per second were varied. Results are tabulated below.

Messages per Connection	Connections	Average Messages per Minute	Average Bytes per Minute
2	150	2044	10928
5	50	3171	16995
10	30	3357	18000
10	10	3530	18926

### 3.3 POPA3D

POPA3D testing was met with some resistance. The initial installation met with the same problems as Postfix. The system was running, and receiving connections, but e-mail was unable to be downloaded. This was due to the precompiled options POPA3D has which determine where mailboxes are stored. Mailboxes were contained in the user home directory, but POPA3D was set to read e-mail from the /var/mail directory. After this option was changed and POPA3D recompiled, the test client downloaded e-mail perfectly. Testing then began, but were met with mixed results.

During the first round of testing, message download numbers on the order of 13,000-15,000 messages per minute were observed, equaling approximately 80 megabytes per minute. Then nondescript errors started appearing. A search of the configuration header file turned up several options that limit the number of children processes that POPA3D will allow for incoming connections. It appeared the system was being flooded with so many requests and POPA3D was refusing to answer them because of these options. These options were adjusted and tested again. The results were lower, but fewer errors were received. The new tests resulted in approximately

3,000 messages per minute. Other POP3 servers may have better results, but there wasn't enough time to configure and test them. Those options are left open for future research and development.

### **3.4 Comments on Results**

While the results were excellent on some tests and mediocre on others, the reader must remember that these results will be better in any actual network. Part of the simulation required that there be a way to compare the outdated hardware used in the prototype to newer hardware likely to be used in any real network. To achieve this several benchmarks were run on the test systems as well as several higher end servers available at the time. All tests were conducted on FreeBSD, and were designed such that they tested the core subsystems involved in the prototype. In this case, CPU performance, tied with memory performance in some tests, and network performance were benchmarked. While benchmark results do not tell the entire picture, in some cases the two-fold performance increase of the benchmark on the higher end systems would be close to that expected of any server application running on them. The table of performance results is in Appendix G.

## **4. Financial Budget**

The objective of this prototype was to create a high performing yet low cost system. The cost descriptions given in the Final Design Report from last semester were for a major overhaul of the entire network, including construction of the new RealServer farm for streaming media. However, since the main focus of the final prototype was on the redesign of Attila, the budget for our prototype for the new distributed Attila is presented.

The group members already owned the majority of the hardware used to build this distributed system. Most of the items purchased were parts needed to complete building the machines, since it was mostly spare parts owned. Other items, such as the Cisco switch, were borrowed. A KVM (keyboard, video, mouse) switch was purchased to make it possible to use one keyboard, monitor, and mouse for all of the machines.

ITEM	QUANTITY	UNIT PRICE	TOTAL
Evercase ATX case	1	\$38.00	\$38.00
Premier ATX case	1	\$25.00	\$25.00
KVM switch	1	\$49.00	\$49.00
KVM cables	4	\$6.00	\$24.00
3.5" floppy drives	3	\$8.00	\$24.00
FIC SD11 Slot A motherboard	1	\$34.99	\$34.99
<b>SUBTOTAL</b>			\$194.99
<b>SHIPPING</b>			\$46.38
<b>TOTAL</b>			\$241.37

## 5. Project Schedule

The Gantt chart the group created for the Final Design Report at the end of last semester was based on the idea of building a mock network to be used by the campus community. This mock network was to be used to gather networking statistics in order to create a sample of the current network's usage. Since the project changed considerably, the Gantt chart presented in the Interim Report on March 17, 2003 contains a more accurate representation of the actual schedule followed. See the attached Gantt chart for the final project schedule.

### III. Conclusion

The final design that was implemented was a redesign of Attila, the main Stevens campus server. The prototype consisted of several servers designed to separate the many services currently being performed by Attila, with the intention of increasing system performance, reliability, robustness, and future expandability.

The premise behind this prototype is high performance and reliability with low cost. It was built using relatively old hardware owned by the group members and a few extra parts purchased in order to make complete systems. A freely available open source operating system (FreeBSD) and freely available open source networking applications (postfix, thttpd, popa3d, etc.) were used to keep costs down.

As shown by the results above, a large percentage of Attila's current load can be tackled with the prototype system, even though the entire system is overshadowed by Attila's sophistication as a computing system. This proves that a distributed system can perform up to the level of one high-powered computer for certain tasks. Based on the results, it was shown that the project was a success in demonstrating high performance and reliability at low cost.

Further testing on the prototype system could not be performed due to time constraints. Should further testing be performed, many things could be done to improve the system's performance. Additional operating system optimizations such as those described above would certainly increase the system's performance. Having more RAM to address the low memory conditions described above would also show a boost. Another less viable (and more expensive) option would be to use more powerful hardware combined with the optimizations just mentioned to show just how powerful the entire system could become. Should all the resources be available,

a large, encompassing test with every server doing its job could be performed with real users and real traffic. This would add to the verification of the data obtained from the prototype system.

## IV. References

1. Stevens, Richard. TCP/IP Illustrated Volume 1: The Protocols. Addison Wesley, 1994.
2. Garfinkel, Simson; Spafford, Gene. Practical Unix & Internet Security. O'Reilly & Associates, Inc., 1996
3. Slatter, Terry; Burton, Bill. Advanced Routing in Cisco Networks. McGraw Hill, 1999.
4. Scambray, Joel; McClure, Stuart; Kurts, George. Hacking Exposed Second Edition. McGraw Hill, 2001.
5. Comer, Douglas E. Internetworking with TCP/IP: Principles, Protocols, and Architectures. Prentice Hall, 2000.
6. Callaghan, Brent. NFS Illustrated. Addison Wesley, 2000.
7. Perlman, Radia. Interconnections Second Edition: Bridges, Routers, Switches, and Internetworking Protocols. Addison Wesley, 2000.
8. Hulton, David. Practical Exploitation of RC4 Weaknesses in WEP Environments.  
[ONLINE]<http://www.dachb0den.com/projects/bsd-airtools/wepexp.txt>
9. Fluhrer, S.; Mantin, I.; Shamir A. Weaknesses in the Key Scheduling Algorithm of RC4.  
[ONLINE] <http://citeseer.nj.nec.com/486392.html>
10. Chase, Jeffrey S.; Gallatin, Andrew J.; Yocum, Kenneth G. End-System Optimizations for High-Speed TCP.  
[ONLINE] [www.cs.duke.edu/ari/publications/endsystem.pdf](http://www.cs.duke.edu/ari/publications/endsystem.pdf)
11. <http://www.postfix.org/>
12. <http://www.acme.com/>
13. <http://www.freebsd.org/>
14. <http://marc.theaimsgroup.com/>

## Appendix A -- FreeBSD Scalability Options

In this appendix a discussion of several kernel and user land options to FreeBSD will be explored that allow it to perform extremely well in high load environments. Like most operating systems, the factory defaults are not designed for the highest of load environments. Most settings are conservative since most systems require operation on even modest hardware. Often the tuning necessary to obtain a system capable of sustaining high loads would exhaust the physical resources of a modest system and potentially cause problems.

In this case, the biggest resource consideration is physical memory. The test systems were equipped with 256 Megabytes of SDRAM each. While this seems like a small amount of memory by today's standards, it is quite large considering the vintage of the tests systems. Given the loads expected on these systems, 256 Megabytes of memory is more than sufficient. The experimental results supported the initial conclusions about the amount of memory required. An even greater number of sustainable clients/connections could have been potentially realized had a larger amount of system memory were used, but in that case the results would have only been better.

The systems needed a larger amount of memory allocated to the kernel because of the large numbers of network connections and files being handled. Each network connection in any operating system requires buffer space to handle the data that has not yet been sent or received to and from the network. In a FreeBSD environment each connection uses a small amount of memory to track the connection as well as the amount of memory each send and receive buffer is set to. Buffer space on FreeBSD is linked to window sizes used in TCP sessions. Increasing the buffer space allows the system to buffer more data for the connection and therefore send as much

data over the connection as possible. Increasing these buffer sizes on slow network links can enhance performance and scalability.

Other parameters that affect server scalability include the numbers of network sockets the system will allow at any given moment, the number of open files, the number of possible processes, accept filters and compiler options. Each are documented here in some detail starting with the kernel level memory options.

### Kernel Memory Options:

```
options    VM_KMEM_SIZE="(10*1024*1024)"
options    VM_KMEM_SIZE_MAX="(100*1024*1024)"
options    VM_KMEM_SIZE_SCALE="4"
```

From /sys/i386/conf/LINT in the FreeBSD source code tree:

```
"# Tune the kernel malloc area parameters.  VM_KMEM_SIZE represents the
# minimum, in bytes, and is typically (12*1024*1024) (12MB).
# VM_KMEM_SIZE_MAX represents the maximum, typically 200 megabytes.
# VM_KMEM_SIZE_SCALE can be set to adjust the auto-tuning factor, which
# typically defaults to 4 (kernel malloc area size is physical memory
# divided by the scale factor)."
```

These are some options that allow a system administrator to allocate an amount of physical memory for kernel usage. In order to support the various kernel data structures that require larger amounts of memory in high load systems the kernel must be given memory to work with. More specific information can be found here:

[<http://docs.freebsd.org/cgi/getmsg.cgi?fetch=271588+0+archive/2001/freebsd-net/20010805.freebsd-net>]

Jasper Wallace wrote:

```
>
> On Fri, 3 Aug 2001, Andre Oppermann wrote:
>
>> Hello guys
>>
>> have got a small problem. I'm running a secondary DNS server for the
>> ccTLD .ch here in Switzerland.
>
>> # vmstat -m
>> Memory statistics by type                Type Kern
```

```

>>   Type InUse MemUse HighUse Limit Requests Limit Limit Size(s)
>>   ...
>>   routetbl607857 85480K 85480K 85480K 2420956 0 0
>> 16,32,64,128,256
>>   ...
>> Memory Totals: In Use Free Requests
>>           91073K 2948K 786316696
>
> If you want to wack up the amount of ram used for the routing table you can
> adjust the amount of ram the kernel uses for it's stuff with something like
> this in your kernel config file:
>
> # 1/2 RAM for the kernel - lets us have full routes.
> options      VM_KMEM_SIZE_SCALE="(2)"
>
> If anyone knows a better way to persuade the kernel to use more space for
> the routing table i'd love to know.

```

On some machines that do routing and have a full view in the kernel I do this in /boot/loader.conf:

```
kern.vm.kmem.size="128000000" # Sets the size of kernel memory
(bytes)
```

But this is not very optimal because only half of the kernel memory can be used for the routing table. The other half (64MB here) is mostly empty. I'll like to know if there is a way to adjust this ratio in the kernel so I get 96MB for the routing table and 32MB for the kernel itself.

--  
Andre

To Unsubscribe: send mail to majordomo@FreeBSD.org  
with "unsubscribe freebsd-net" in the body of the message

## Network Options:

Usually one method of increasing kernel memory is sufficient. The system defaults to about 12 megabytes which, at the send and receive network buffer sizes used, is insufficient. An explanation of a related option, NMBCLUSTERS will help show why. From the tuning manual page:

```
"kern.ipc.nmbclusters may be adjusted to increase the number of network mbufs the system is willing to allocate. Each cluster represents approximately 2K of memory, so a value of 1024 represents 2M of kernel memory reserved for network buffers. You can do a simple calculation to figure out how many you need. If you have a web server which maxes out at 1000 simultaneous connections, and each connection eats a 16K receive and 16K send buffer, you need approximately 32MB worth of network buffers to deal with it. A good rule of thumb is to multiply by 2, so 32MBx2 = 64MB/2K = 32768. So for this case you would want to set
```

kern.ipc.nmbclusters to 32768. We recommend values between 1024 and 4096 for machines with moderates amount of memory, and between 4096 and 32768 for machines with greater amounts of memory. Under no circumstances should you specify an arbitrarily high value for this parameter, it could lead to a boot-time crash. The -m option to netstat(1) may be used to observe network cluster use. Older versions of FreeBSD do not have this tunable and require that the kernel config(8) option NMBCLUSTERS be set instead."

As seen in the above description, the kernel memory required to support the above number of network buffer clusters is 32 megabytes. This is for the case described above. Considering the network buffer space in the prototype was 32K send and 57K receive, approximately 90 megabytes of kernel memory was needed for just network connections. This does not include memory needed for file handling, caching, and any other internal data structures. This also does not include any user space application memory requirements. As can be seen the largest restriction in any high load system is memory.

Another restriction is the number of files the system can have open at any given time. This is a problem on Linux, FreeBSD and Solaris as well as other Unix operating systems. In these operating systems certain primitives are considered files such as sockets, pipes, and physical files. In FreeBSD the problem is easily remedied, but it must be done in certain locations otherwise the full effectiveness is not realized.

```
options      MAXFILES=32768
```

This kernel option in FreeBSD is used to augment the number of files capable of being open at any given time. Another option:

```
maxusers     256
```

is available and the two are linked; however, maxusers is used to change many kernel internals. In a server that sees large user interaction (shell server) this may be a better solution to

adjusting the tolerances within the system than only adjusting MAXFILES. However, for network oriented service systems (HTTP, POP, SMTP, etc.) it is better to adjust the MAXFILES option as it leads to less kernel bloat since the option is tied into fewer kernel structures. This option, to be fully effective, needs to be implemented in either the kernel configuration file before a kernel compilation, or before the kernel is booted. To implement the MAXFILES as well as some of the VM\_KMEM options before kernel booting, they must be placed in /boot/loader.conf with proper syntax requirements in mind. The MAXFILES options can also be set with FreeBSD's sysctl system; however, to affect network primitives they must be set in the kernel or in /boot/loader.conf because of very early kernel initialization that uses their values. The MAXFILES values are used to initialize several network structures very shortly after the kernel starts loading. If they are set with the sysctl controls after the system boots the change will affect actual file numbers allowed to be open, but not network primitives.

### **Accept Filters:**

The best descriptions of Accept Filters and their performance have already been written. In order to save time they are referenced below from two locations. The first is from the FreeBSD manual page `accf_http(9)`:

"This is a filter to be placed on a socket that will be using `accept()` to receive incoming HTTP connections.

It prevents the application from receiving the connected descriptor via `accept()` until either a full HTTP/1.0 or HTTP/1.1 HEAD or GET request has been buffered by the kernel.

If something other than a HTTP/1.0 or HTTP/1.1 HEAD or GET request is received the kernel will allow the application to receive the connection descriptor via `accept()`.

The utility of `accf_http(9)` is such that a server will not have to context switch several times before performing the initial parsing of the request. This effectively reduces the amount of required CPU utilization to handle incoming requests by keeping active processes in preforking servers such as Apache low and reducing the size of the filedescriptor set that needs to be managed by interfaces such as `select()`, `poll()` or `kevent()` based servers.

The `accf_http(9)` kernel option is also a module that can be enabled at runtime via `kldload(8)` if the `INET` option has been compiled into the kernel."

A second useful source of information on accept filters comes from a subsection of the Apache web server tuning page:

[<http://httpd.apache.org/docs/misc/perf-bsd44.html>]

"Versions of FreeBSD from August 2000 onwards include a feature called "accept filters" which delay the return from `accept()` until a condition has been met, e.g. an HTTP request has arrived. This postpones the requirement for a child process to handle the new connection which therefore increases the number of connections that a given number of child processes can handle. It also allows a child process to accomplish more immediately after `accept()` returns (because the request is already available to be read) so there is less context switching.

Accept filters provide the most benefit on servers that are already so busy that they are configured with "KeepAlive Off". HTTP KeepAlive (aka persistent connections) avoids the cost of setting up a new connection for every request, but connections that are being kept alive use up one of the available child processes. Since there is a limited number of child processes this can significantly reduce the capacity of the server. The viewers of a web site will still get a lot of the benefit of persistent connections even with a very small `KeepAliveTimeout` so you should try reducing it before turning it off altogether.

To enable accept filtering, you must either load the appropriate accept filter module, e.g. with the command `kldload accf_http`, or compile a kernel with options `ACCEPT_FILTER_HTTP`. Apache will then enable filtering when it is restarted.

Accept filters are compiled in if the symbol `SO_ACCEPTFILTER` is defined on the machine on which Apache is built. Additionally there is a directive `AcceptFilter` to switch the filters on or off. The default is on; except when apache is compiled with `-D AP_ACCEPTFILTER_ON`.

See the manual page `accf_http(9)` for more information."

Fortunately the Apache web server is not the only server that utilizes accept filters. `httpd` also has support for them as well as `kqueue()`. Accept filters and `kqueue()` are some of the advanced architectural features of FreeBSD that allow it a performance advantage over other operating systems for high load serving environments.

### **Compiler Options:**

Another benefit of open source operating systems is the availability of various compilers. In closed source systems you do not have the benefit of recompiling the operating system itself

with more aggressive compiler options nor the applications to be installed on the system. With the open source systems this becomes a good area for focus on performance. The most commonly used compiler in open source systems is the Gnu Compiler Collection (GCC). For testing purposes platform specific options were used in the compiler to get a boost in performance. For instance, when compiling the operating system kernel, unneeded drivers and subsystems are stripped out and GCC is directed to use its best optimization routines.

Some of these optimization routines are based on processor class. For instance, the Intel x86 Pentium Pro and Pentium II/III/4 processors are classified as an i686 (-march=i686, -mcpu=i686 in GCC). Others are general optimizations (-O2 in GCC) that result in more efficient compiled byte code. Sometimes these optimizations can lead to instability in the system. In this case, such a problem was encountered with some of the benchmarking routines run. Backing the optimizations off to lower levels leads to a properly running binary. In these cases it is up to the system administrator to properly test the program with various compiler options to determine a performance increase (if any) as well as any stability issues that may be introduced by their use. For the server daemons used, no such stability issues were encountered but no specific tests for higher performance using the optimizations were performed.

Using newer versions of GCC, an administrator can extract an even greater performance boost for applications and system. The base GCC included in FreeBSD 4.7, the test platform used in the prototype, is 2.54. In FreeBSD 5.0 the default GCC version is from the 3.2 series. [<http://gcc.gnu.org/gcc-3.0/features.html>] Another option for Intel based systems is to use Intel's native compiler, ICC. Benchmarks between GCC and ICC show that in several cases ICC has a decided advantage over GCC on Pentium 4 systems. [[http://www.coyotegulch.com/reviews/intel\\_comp/intel\\_gcc\\_bench2.html](http://www.coyotegulch.com/reviews/intel_comp/intel_gcc_bench2.html)]

## **Appendix B -- FreeBSD Usability and Ports**

FreeBSD is a BSD-UNIX operating system based on original code by the University of California at Berkeley. It has a long heritage of high performance and stability. The current incarnations of BSD include FreeBSD, NetBSD, and OpenBSD. Each variant has its strengths. OpenBSD is a multi-platform operating system that strives to achieve excellent security through frequent source code auditing and intelligent programming and design. NetBSD strives to be the world's most ported operating system. It currently is fully supported on 21 platforms, with others in varying stage of operation and support. FreeBSD is the oldest and most mature of the modern BSD operating systems. It strives for performance, stability, and usability on several platforms.

The biggest benefit of using open source operating systems is clear: licensing costs. Typical installations of Windows 2000 Server with a 5 client access license (CAL) cost in the range of \$700-900. This does not include any 3rd party applications required to fully administer the system. Moving to an open source system allows companies and institutions to drive down both operating and setup costs significantly. Common myths about open source software include high cost of operation, difficulty of operation, and lack of support. The truth is, operating costs for these open source systems are usually minor compared to closed source for profit systems; usage and operation in today's world are very easy, and support is abundant from the community involved with the software.

There are several Linux servers in operation in the IT department currently. We feel FreeBSD is a better choice for its easier to manage infrastructure (ports collection), greater stability (depending on application), and higher performance (depending on application). As mentioned above a FreeBSD system is simple to maintain through the ports collection [<http://www.freebsd.org/ports/>]. The ports collection is a framework that allows easy central

software management and easy configuration and building of software packages. Unlike several Linux distributions and the requirement to search for RPM's of library dependencies for a desired software package, software on FreeBSD can be installed with as little as 2 commands. For an example package, the process is as follows:

1. Change to the ported program's directory in the ports collection.  
(cd /usr/ports/www/mozilla)
2. Build the port.  
(make install)

The FreeBSD ports collection does the rest. It will trace the dependencies required for the building of the Mozilla software package. It will download, configure, patch, build, and install those dependencies. Then it will fetch the source code for the Mozilla program, patch, configure, build and install it. Upon installation the "package" is recorded in a central database so that future removal, maintenance, and updates are simple. If the Mozilla program is updated at some point in the future all one needs to do is update the ports collection using CVS or CVSup from the FreeBSD project's website, and perform steps 1 and 2 above. Removal of the old package is not necessary, but can be done to reduce clutter in the package database. If the new Mozilla requires a newer dependency the ports collection will fetch, build and install it without requiring user interaction. If a system administrator does not wish to build the program from source code then there are prebuilt packages available from the FreeBSD project which are continually built by compiler clusters. Again, even if using prebuilt packages, the system will automatically fetch and install dependencies thereby saving the system administrator's time and energy.

This greatly simplifies system maintenance for a system administrator. It saves the administrator the time of finding the required dependencies for a given program, determining what configuration options he or she needs to perform, and tracing down compiler issues on a

given platform. The FreeBSD ports team determines these things in advance and creates patches to make the software compile and run on FreeBSD. The Ports collection is under continual maintenance by members of the team and non-critical updates to the ports tree usually occur within 1-2 days of the release of a new version of a given open source program.

## Appendix C -- tthttpd

tthttpd, the "tiny, turbo, throttling web server" is a high speed HTTP server written by Jef Poskanzer. From tthttpd's website (<http://www.acme.com/software/tthttpd/>):

"tthttpd is a simple, small, portable, fast, and secure HTTP server.

Simple:

It handles only the minimum necessary to implement HTTP/1.1. Well, maybe a little more than the minimum.

Small:

See the comparison chart. It also has a very small run-time size, since it does not fork and is very careful about memory allocation.

Portable:

It compiles cleanly on most any Unix-like OS, specifically including FreeBSD, SunOS 4, Solaris 2, BSD/OS, Linux, OSF.

Fast:

In typical use it's about as fast as the best full-featured servers (Apache, NCSA, Netscape). Under extreme load it's much faster.

Secure:

It goes to great lengths to protect the web server machine against attacks and breakins from other sites.

It also has one extremely useful feature (URL-traffic-based throttling) that no other server currently has. Plus, it supports IPv6 out of the box, no patching required."

Included on the Acme tthttpd website is a good description about the reasons behind some of the design decisions that went into tthttpd:

"Non-blocking I/O is good:

select() / poll() / kqueue() are Unix system calls used to multiplex between a bunch of file descriptors. To understand why this is important we have to go back through the history of web servers.

The basic operation of a web server is to accept a request and send back a response. The first web servers were probably written to do exactly that. Their users no doubt noticed very quickly that while the server was sending a response to someone else, they couldn't get their own requests serviced. There would have been long annoying pauses.

The second generation of web servers addressed this problem by forking off a child process for each request. This is very straightforward to do under Unix, only a few extra lines of code. CERN and NCSA 1.3 are examples of type of server. Unfortunately, forking a process is a fairly expensive operation, so performance of this type of server is still pretty poor. The long random pauses are gone, but instead every request has a short constant pause at startup. Because of this, the server can't handle a high rate of connections.

A slight variant of this type of server uses "lightweight processes" or "threads" instead of full-blown Unix processes. This is better, but there is no standard LWP/threads interface so this approach is inherently non-portable. Examples of these servers: MDMA and phttpd, both of which run only under Solaris 2.x.

The third generation of servers is called "pre-forking". Instead of starting a new subprocess for each request, they have a pool of subprocesses that they keep around and re-use. NCSA 1.4, Apache, and Netscape Netsite are examples of this type. Performance of these servers is excellent, they can handle from two to ten times as many connections per second as the forking servers. One problem, however, is that implementing this simple-to-state idea turns out to be fairly complicated and non-portable. The method used by NCSA involves transferring a file descriptor from the parent process to an already-existing child process; you can hardly use the same code on any two different OS's, and some OS's (e.g. Linux) don't support it at all. Apache uses a different method, with all the child processes doing their own round-robin work queue via lock files, which brings in issues of portability/speed/deadlock. Besides, you still have multiple processes hanging around using up memory and context-switch CPU cycles. Which brings us to...

The fourth generation. One process only. No non-portable threads/LWPs. Sends multiple files concurrently using non-blocking I/O, calling select()/poll()/kqueue() to tell which ones are ready for more data. Speed is excellent. Memory use is excellent. Portability is excellent. Examples of this generation: Spinner, Open Market, and thttpd. Perhaps Apache will switch to this method at some point. I really can't understand why they went with that complicated pre-forking stuff. Using non-blocking I/O is just not that hard."

[<http://www.acme.com/software/thttpd/notes.html#nbio>]

thttpd was used for the simulation and the recommendation for the new campus network due to its astonishing performance, agreeable price (free), and excellent scalability on FreeBSD. thttpd makes use of several FreeBSD native system calls, most notably the kqueue() call which allows it to receive kernel level queuing services thereby greatly increasing performance under high loads. Test trials were conducted with a default configuration of thttpd, although there are several performance options that can be specified at compile time. The following are excerpts from the thttpd website with additional comments:

CGI\_TIMELIMIT

How many seconds to allow CGI programs to run before killing them. This is in case someone writes a CGI program that goes into an infinite loop, or does a massive database lookup that would take hours, or whatever. If you don't want any limit, comment this out, but that's probably a really bad idea.

A typical value is 300.

If campus administrators decided to allow CGI services for student or organization websites this would be a significant means of limiting system usage. Setting this option would ensure that certain CGI programs do not consume all system resources available on the server.

#### CGI\_NICE

nice(2) value to use for CGI programs. If this is left undefined, CGI programs run at normal priority. A typical value, if defined, is 10.

This is also a desirable setting when dealing with CGI applications. It works the same as the standard Unix 'nice' program. This option would set the priority with which CGI applications run on the host system and can be used to prevent a script from monopolizing resources over any other running application.

#### DESIRED\_MAX\_MAPPED\_FILES

The mmap cache tries to keep the total number of mapped files below this number, so you don't run out of kernel file descriptors. If you have reconfigured your kernel to have more descriptors, you can raise this and thttpd will keep more maps cached. However it's not a hard limit, thttpd will go over it if you really are accessing a whole lot of files. A typical value is 2000.

#### DESIRED\_MAX\_MAPPED\_BYTES

The mmap cache also tries to keep the total mapped bytes below this number, so you don't run out of address space. Again it's not a hard limit, thttpd will go over it if you really are accessing a bunch of large files. A typical value is 1000000000.

While the tests performed included a small subset of the Stevens website as test content, these options would be useful if IT administrators were serving large numbers of web pages. These options will adjust the number of files and the total size of files cached by thttpd with mmap(). As it states, the server will increase its cache if it needs to, but having a larger default

will allow handling load spikes with greater efficiency. If the system has to stop to allocate more file cache space its attention will not be focused on serving content to end users.

#### SPARE\_FDS

Number of file descriptors to reserve for uses other than connections. Currently this is 5, representing one for the listen fd, one for dup()ing at connection startup time, one for reading the file, one for syslog, and one possibly for the log file.

In the same spirit as having larger cache sizes to avoid allocation of memory to file caching when the system absolutely needs it, this option will allocate spare file descriptors (fd). Raising this option would force thttpd to reserve extra file descriptors for later use so that it does not need to stop to request them when needed.

As the test results show, thttpd is a robust HTTP server. It is scalable to a large number of simultaneous users even on modest hardware, it has configurable throttling capabilities to limit the traffic in use for a given file or URL, it is a free software solution, and it is a secure means of serving HTTP data. Since a large number of Stevens web pages as well as those of students do not use CGI applications or dynamic data-driven languages like PHP thttpd is recommended for each of its qualities outlined above and proven by the simulation performed.

## Appendix D -- Postfix

Postfix is a mailer that was started as an alternative to the widely used Sendmail program.

From the postfix website: (<http://www.postfix.org>)

Postfix attempts to be fast, easy to administer, and secure, while at the same time being sendmail compatible enough to not upset existing users. Thus the outside has a sendmail-ish flavor, but the inside is completely different.

This software was formerly known as VMailer. It was released by the end of 1998 as the IBM Secure Mailer. From then on it has lived on as Postfix.

The Postfix website listed the goals of the mail program. These included some of the reasons this program was chosen.

- *Wide dissemination. Postfix must be adopted by lots of people in order to make a significant impact on Internet mail performance and security. Therefore the software is given away for free, with no strings attached to it.*
- *Performance. Postfix is up to three times as fast as its nearest competitor. A desktop PC running Postfix can receive and deliver a million different messages per day. Postfix uses web server tricks to reduce process creation overhead and uses other tricks to reduce file system overhead, without compromising reliability.*
- *Compatibility. Postfix is designed to be sendmail-compatible to make migration easy. Postfix supports /var/spool/mail, /etc/aliases, NIS, and ~/.forward files. However, Postfix also attempts to be easy to administer, and therefore it does not use sendmail.cf.*
- *Safety and robustness. Postfix is designed to behave rationally under stress. When the local system runs out of disk space or memory, the Postfix software backs off, instead of making the problem worse. By design, no Postfix program keeps growing as the number of messages etc. increases. Postfix is designed to stay in control.*
- *Flexibility. Postfix is built from over a dozen little programs that each perform only one specific task: receive a message via SMTP, deliver a message via SMTP, deliver a message locally, rewrite an address, and so on. Sites with specific requirements can replace one or more little programs by alternative versions. And it is easy to disable functionality, too: firewalls and client workstations don't need local delivery at all.*
- *Security. Postfix uses multiple layers of defense to protect the local system against intruders. Almost every Postfix daemon can run in a chroot jail with fixed low privileges. There is no direct path from the network to the security-sensitive local delivery programs - an intruder has to break through several other programs first. Postfix does not even trust the contents of its own queue files, or the contents of its own IPC messages. Postfix filters sender-provided information before exporting it via environment variables. Last but not least, no Postfix program is set-uid.*

[<http://www.postfix.org/goals.html>]

The goals listed on the website generally sum up the reasons for its use in the simulation. They will be discussed in the following text. The first goal is very useful in an educational environment. Making the program free for use clears up funds for other projects or for use directly for the benefit of students. As can be seen in many websites and articles, the software is widely used in the server community. This provides a large community for instruction and communication about troubleshooting or performance issues.

The heightened performance can be explained by the architecture of the postfix program. The following is from the postfix website [<http://www.postfix.org/architecture.html>]

*“Postfix is based on semi-resident, mutually-cooperating, processes that perform specific tasks for each other, without any particular parent-child relationship. Again, doing work in separate processes gives better insulation than using one big program. In addition, the Postfix approach has the advantage that a service such as address rewriting is available to every Postfix component program, without incurring the cost of process creation just to rewrite one address. By the way: I do not claim that Postfix is the only (mail) program using this approach. Even in this relatively young discipline it is hard to come up something new that no-one ever did before.*”

*Postfix is implemented as a resident master server that runs Postfix daemon processes on demand: daemon processes to send or receive network mail messages, daemon processes to deliver mail locally, etc. These processes are created up to a configurable number, are re-used for a configurable number of times, and go away after a configurable amount of idle time. This approach drastically reduces process creation overhead while still providing the good insulation from separate processes.*

*Postfix is intended to be a Sendmail replacement. For this reason it tries to be compatible with existing infrastructure. However, many parts of the Postfix system, such as the local delivery program, are easily replaced by editing an inetd-like configuration file. For example, the plan is to provide an alternate local delivery program that runs at a fixed low privilege, for POP/IMAP users that never log into the shell, and that may not even have a UNIX account.*

*As a result of this architecture, Postfix is easy to strip down to the bare minimum. Subsystems that are turned off cannot be exploited. Firewalls do not need local delivery. On client workstations, one disables both the smtp listener and local delivery subsystems; or the client mounts the maildrop directory from a file server, and runs no resident Postfix processes at all.”*

The compatibility, safety, flexibility, and security also provide valid reasons for using Postfix. For example, in case a given server function was only compatible with sendmail, the

sendmail compatibility in Postfix would resolve that problem. The software would be able to operate as if sendmail was running on the system. An argument against using sendmail to begin with is administration. Postfix is configured using a main.cf file. This is much easier to configure and maintain than the sendmail.cf configuration file.

Having a highly configurable system allowed for setting up options specifically targeted for optimum use in any type of network desired. The ability to use alternative programs for specific tasks leaves room for more avenues in the design process. These are just some of the benefits of using Postfix as the MTA in a given network. Some of the important options are below.

```
myhostname = server1.seniord.net
mydomain = seniord.net
```

This is the basic setup of the host and domain name. For the simulations performed, the host was given server1 as the name in the domain seniord.net.

```
# SENDING MAIL
# The myorigin parameter specifies the domain that locally-posted
# mail appears to come from.
myorigin = $myhostname
```

Locally posted mail will appear to come from \$myhostname, which is “server1.seniord.net” from the entry above.

```
# RECEIVING MAIL
# The inet_interfaces parameter specifies the network interface
# addresses that this mail system receives mail on. By default,
# the software claims all active interfaces on the machine. The
# parameter also controls delivery of mail to user@[ip.address].
inet_interfaces = all
```

This entry will allow the system to claim all the mail that comes to its network address.

These include server1.seniord.net and localhost.

```
# The mydestination parameter specifies the list of domains that this
# machine considers itself the final destination for.
#
mydestination = $myhostname, localhost.$mydomain, $mydomain
```

In this configuration example, the mail system will accept mail for the following:

```
server1.seniord.net, localhost.seniord.net, and seniord.net
```

The following option will allow the administrator to easily manage trusted clients and relay control. Trusted clients can be specified by IP, subnet or class. In this case relaying was allowed for all 3 classes. Of course, in a real world environment, this would be restricted to authorized SMTP clients.

```
# TRUST AND RELAY CONTROL
```

```
# The mynetworks parameter specifies the list of "trusted" SMTP
# clients that have more privileges than "strangers".
#
# In particular, "trusted" SMTP clients are allowed to relay mail
# through Postfix. See the smtpd_recipient_restrictions parameter
# in file sample-smtpd.cf.
#
mynetworks = 192.168.0.0/24, 10.0.0.0/24, 127.0.0.0/8
```

## Appendix E – FreeBSD Kernel Configuration Files

What follows is the configuration file used for SERVER1 and SERVER2.

---

```
#
# GENERIC -- Generic kernel configuration file for FreeBSD/i386
#
# For more information on this file, please read the handbook section on
# Kernel Configuration Files:
#
#   http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/kernelconfig-
#   config.html
#
# The handbook is also available locally in /usr/share/doc/handbook
# if you've installed the doc distribution, otherwise always see the
# FreeBSD World Wide Web server (http://www.FreeBSD.org/) for the
# latest information.
#
# An exhaustive list of options and more detailed explanations of the
# device lines is also present in the ./LINT configuration file. If you are
# in doubt as to the purpose or necessity of a line, check first in LINT.
#
# $FreeBSD: src/sys/i386/conf/GENERIC,v 1.246.2.48 2002/08/31 20:28:26 obrien
Exp $

machine      i386
#cpu         I386_CPU
#cpu         I486_CPU
#cpu         I586_CPU
cpu          I686_CPU
ident        SERVER1
maxusers     0

#makeoptions  DEBUG=-g      #Build kernel with gdb(1) debug symbols

#options      MATH_EMULATE   #Support for x87 emulation
options      INET            #InterNETworking
#options      INET6          #IPv6 communications protocols
options      FFS             #Berkeley Fast Filesystem
options      FFS_ROOT        #FFS usable as root device [keep this!]
options      SOFTUPDATES     #Enable FFS soft updates support
options      UFS_DIRHASH     #Improve performance on big directories
#options      MFS            #Memory Filesystem
#options      MD_ROOT        #MD is a potential root device
options      NFS             #Network Filesystem
#options      NFS_ROOT       #NFS usable as root device, NFS required
options      MSDOSFS         #MSDOS Filesystem
options      CD9660          #ISO 9660 Filesystem
#options      CD9660_ROOT    #CD-ROM usable as root, CD9660 required
options      PROCFS          #Process filesystem
```

```

options      COMPAT_43          #Compatible with BSD 4.3 [KEEP THIS!]
options      SCSI_DELAY=15000  #Delay (in ms) before probing SCSI
#options     UCONSOLE          #Allow users to grab the console
#options     USERCONFIG        #boot -c editor
#options     VISUAL_USERCONFIG #visual boot -c editor
options     KTRACE             #ktrace(1) support
options     SYSVSHM            #SYSV-style shared memory
options     SYSVMSG            #SYSV-style message queues
options     SYSVSEM            #SYSV-style semaphores
options     P1003_1B          #Posix P1003_1B real-time extensions
options     _KPOSIX_PRIORITY_SCHEDULING
#options     ICMP_BANDLIM      #Rate limit bad replies
options     KBD_INSTALL_CDEV   # install a CDEV entry in /dev
options     AHC_REG_PRETTY_PRINT # Print register bitfields in debug
                                     # output. Adds ~128k to driver.
options     AHD_REG_PRETTY_PRINT # Print register bitfields in debug
                                     # output. Adds ~215k to driver.

options     ACCEPT_FILTER_DATA
options     ACCEPT_FILTER_HTTP
options     NMBCLUSTERS=32768
options     VM_KMEM_SIZE="(192*1024*1024)"
#options     VM_KMEM_SIZE_SCALE="2"
options     MAXFILES=32768

# To make an SMP kernel, the next two are needed
#options     SMP                # Symmetric MultiProcessor Kernel
#options     APIC_IO            # Symmetric (APIC) I/O

Device       isa
#device     eisa
device      pci

# Floppy drives
device      fdc0  at isa? port IO_FD1 irq 6 drq 2
device      fd0   at fdc0 drive 0
device      fd1   at fdc0 drive 1
#
# If you have a Toshiba Libretto with its Y-E Data PCMCIA floppy,
# don't use the above line for fdc0 but the following one:
#device      fdc0

# ATA and ATAPI devices
device      ata0  at isa? port IO_WD1 irq 14
device      ata1  at isa? port IO_WD2 irq 15
device      ata
device      atadisk # ATA disk drives
device      atapicd # ATAPI CDROM drives
#device     atapifd # ATAPI floppy drives
#device     atapist # ATAPI tape drives
options     ATA_STATIC_ID #Static device numbering

# SCSI Controllers
#device     ahb          # EISA AHA1742 family
device     ahc           # AHA2940 and onboard AIC7xxx devices
#device     ahd          # AHA39320/29320 and onboard AIC79xx devices

```

```

#device    amd          # AMD 53C974 (Tekram DC-390(T))
#device    isp          # Qlogic family
#device    mpt          # LSI-Logic MPT/Fusion
#device    ncr          # NCR/Symbios Logic
#device    sym          # NCR/Symbios Logic (newer chipsets)
#options   SYM_SETUP_LP_PROBE_MAP=0x40
                # Allow ncr to attach legacy NCR devices when
                # both sym and ncr are configured

#device    adv0        at isa?
#device    adw
#device    bt0         at isa?
#device    aha0        at isa?
#device    aic0        at isa?

#device    ncv          # NCR 53C500
#device    nsp          # Workbit Ninja SCSI-3
#device    stg          # TMC 18C30/18C50

# SCSI peripherals
device     scbus        # SCSI bus (required)
device     da           # Direct Access (disks)
device     sa           # Sequential Access (tape etc)
device     cd           # CD
device     pass         # Passthrough device (direct SCSI access)

# RAID controllers interfaced to the SCSI subsystem
#device    asr          # DPT SmartRAID V, VI and Adaptec SCSI RAID
#device    dpt          # DPT Smartcache - See LINT for options!
#device    iir          # Intel Integrated RAID
#device    mly          # Mylex AcceleRAID/eXtremeRAID
#device    ciss         # Compaq SmartRAID 5* series

# RAID controllers
#device    aac          # Adaptec FSA RAID, Dell PERC2/PERC3
#device    aacp         # SCSI passthrough for aac (requires CAM)
#device    ida          # Compaq Smart RAID
#device    amr          # AMI MegaRAID
#device    mlx          # Mylex DAC960 family
#device    twe          # 3ware Escalade

# atkbd0 controls both the keyboard and the PS/2 mouse
device     atkbd0       at isa? port IO_KBD
device     atkbd0       at atkbd? irq 1 flags 0x1
device     psm0         at atkbd? irq 12
device     vga0         at isa?

# splash screen/screen saver
pseudo-device splash

# syscons is the default console driver, resembling an SCO console
device     sc0          at isa? flags 0x100

# Enable this and PCVT_FREEBSD for pcvt vt220 compatible console driver
#device    vt0         at isa?
#options   XSERVER          # support for X server on a vt console
#options   FAT_CURSOR       # start with block cursor

```

```

# If you have a ThinkPAD, uncomment this along with the rest of the PCVT
lines
#options      PCVT_SCANSET=2      # IBM keyboards are non-std

# Floating point support - do not disable.
device        npx0   at nexus? port IO_NPX irq 13

# Power management support (see LINT for more options)
#device        apm0   at nexus? disable flags 0x20 # Advanced Power Management

# PCCARD (PCMCIA) support
#device        card
#device        pcic0  at isa? irq 0 port 0x3e0 iomem 0xd0000
#device        pcic1  at isa? irq 0 port 0x3e2 iomem 0xd4000 disable

# Serial (COM) ports
device        sio0   at isa? port IO_COM1 flags 0x10 irq 4
device        sio1   at isa? port IO_COM2 irq 3
#device        sio2   at isa? disable port IO_COM3 irq 5
#device        sio3   at isa? disable port IO_COM4 irq 9

# Parallel port
#device        ppc0   at isa? irq 7
#device        ppbus          # Parallel port bus (required)
#device        lpt           # Printer
#device        plip          # TCP/IP over parallel
#device        ppi           # Parallel port interface device
#device        vpo           # Requires scbus and da

# PCI Ethernet NICs.
#device        de           # DEC/Intel DC21x4x ('`Tulip'`)
#device        em           # Intel PRO/1000 adapter Gigabit Ethernet Card
('`Wiseman'`)
#device        txp          # 3Com 3cR990 ('`Typhoon'`)
#device        vx           # 3Com 3c590, 3c595 ('`Vortex'`)

# PCI Ethernet NICs that use the common MII bus controller code.
# NOTE: Be sure to keep the 'device miibus' line in order to use these NICs!
device        miibus       # MII bus support
#device        dc           # DEC/Intel 21143 and various workalikes
device        fxp          # Intel EtherExpress PRO/100B (82557, 82558)
#device        pcn          # AMD Am79C97x PCI 10/100 NICs
#device        rl           # RealTek 8129/8139
#device        sf           # Adaptec AIC-6915 ('`Starfire'`)
#device        sis          # Silicon Integrated Systems SiS 900/SiS 7016
#device        ste          # Sundance ST201 (D-Link DFE-550TX)
#device        tl           # Texas Instruments ThunderLAN
#device        tx           # SMC EtherPower II (83c170 ``EPIC'`)
#device        vr           # VIA Rhine, Rhine II
#device        wb           # Winbond W89C840F
device        xl           # 3Com 3c90x ('`Boomerang'`, ``Cyclone'`)
#device        bge          # Broadcom BCM570x ('`Tigon III'`)

# ISA Ethernet NICs.
# 'device ed' requires 'device miibus'
#device        ed0         at isa? port 0x280 irq 10 iomem 0xd8000

```

```

#device      ex
#device      ep
#device      fe0    at isa? port 0x300
# Xircom Ethernet
#device      xe
# PRISM I IEEE 802.11b wireless NIC.
#device      awi
# WaveLAN/IEEE 802.11 wireless NICs. Note: the WaveLAN/IEEE really
# exists only as a PCMCIA device, so there is no ISA attachment needed
# and resources will always be dynamically assigned by the pccard code.
#device      wi
# Aironet 4500/4800 802.11 wireless NICs. Note: the declaration below will
# work for PCMCIA and PCI cards, as well as ISA cards set to ISA PnP
# mode (the factory default). If you set the switches on your ISA
# card for a manually chosen I/O address and IRQ, you must specify
# those parameters here.
#device      an
# The probe order of these is presently determined by i386/isa/isa_compat.c.
#device      ie0    at isa? port 0x300 irq 10 iomem 0xd0000
#device      le0    at isa? port 0x300 irq 5 iomem 0xd0000
#device      lnc0   at isa? port 0x280 irq 10 drq 0
#device      cs0    at isa? port 0x300
#device      sn0    at isa? port 0x300 irq 10

# Pseudo devices - the number indicates how many units to allocate.
pseudo-device loop      # Network loopback
pseudo-device ether     # Ethernet support
pseudo-device sl       1 # Kernel SLIP
pseudo-device ppp       1 # Kernel PPP
pseudo-device tun       # Packet tunnel.
pseudo-device pty       # Pseudo-ttys (telnet etc)
#pseudo-device md       # Memory "disks"
#pseudo-device gif      # IPv6 and IPv4 tunneling
#pseudo-device faith 1  # IPv6-to-IPv4 relaying (translation)

# The `bpf' pseudo-device enables the Berkeley Packet Filter.
# Be aware of the administrative consequences of enabling this!
pseudo-device bpf       #Berkeley packet filter

# USB support
#device      uhci      # UHCI PCI->USB interface
#device      ohci      # OHCI PCI->USB interface
#device      usb       # USB Bus (required)
#device      ugen      # Generic
#device      uhid      # "Human Interface Devices"
#device      ukbd      # Keyboard
#device      ulpt      # Printer
#device      umass     # Disks/Mass storage - Requires scbus and da
#device      ums       # Mouse
#device      uscanner  # Scanners
#device      urio      # Diamond Rio MP3 Player
# USB Ethernet, requires mii
#device      aue       # ADMtek USB ethernet
#device      cue       # CATC USB ethernet
#device      kue       # Kawasaki LSI USB Ethernet

```

## Appendix F – Postfix Configuration File (Main.cf)

```
# Global Postfix configuration file. This file lists only a subset
# of all 250+ parameters. See the sample-xxx.cf files for a full list.
#
# The general format is lines with parameter = value pairs. Lines
# that begin with whitespace continue the previous line. A value can
# contain references to other $names or ${name}s.
#
# NOTE - CHANGE NO MORE THAN 2-3 PARAMETERS AT A TIME, AND TEST IF
# POSTFIX STILL WORKS AFTER EVERY CHANGE.

# SOFT BOUNCE
#
# The soft_bounce parameter provides a limited safety net for
# testing. When soft_bounce is enabled, mail will remain queued that
# would otherwise bounce. This parameter disables locally-generated
# bounces, and prevents the SMTP server from rejecting mail permanently
# (by changing 5xx replies into 4xx replies). However, soft_bounce
# is no cure for address rewriting mistakes or mail routing mistakes.
#
#soft_bounce = no

# LOCAL PATHNAME INFORMATION
#
# The queue_directory specifies the location of the Postfix queue.
# This is also the root directory of Postfix daemons that run chrooted.
# See the files in examples/chroot-setup for setting up Postfix chroot
# environments on different UNIX systems.
#
queue_directory = /var/spool/postfix

# The command_directory parameter specifies the location of all
# postXXX commands.
#
command_directory = /usr/local/sbin

# The daemon_directory parameter specifies the location of all Postfix
# daemon programs (i.e. programs listed in the master.cf file). This
# directory must be owned by root.
#
daemon_directory = /usr/local/libexec/postfix

# QUEUE AND PROCESS OWNERSHIP
#
# The mail_owner parameter specifies the owner of the Postfix queue
# and of most Postfix daemon processes. Specify the name of a user
# account THAT DOES NOT SHARE ITS USER OR GROUP ID WITH OTHER ACCOUNTS
# AND THAT OWNS NO OTHER FILES OR PROCESSES ON THE SYSTEM. In
# particular, don't specify nobody or daemon. PLEASE USE A DEDICATED
# USER.
#
mail_owner = postfix

# The default_privs parameter specifies the default rights used by
```

```
# the local delivery agent for delivery to external file or command.
# These rights are used in the absence of a recipient user context.
# DO NOT SPECIFY A PRIVILEGED USER OR THE POSTFIX OWNER.
#
#default_privs = nobody

# INTERNET HOST AND DOMAIN NAMES
#
# The myhostname parameter specifies the internet hostname of this
# mail system. The default is to use the fully-qualified domain name
# from gethostname(). $myhostname is used as a default value for many
# other configuration parameters.
#
myhostname = server1.seniord.net
#myhostname = virtual.domain.tld

# The mydomain parameter specifies the local internet domain name.
# The default is to use $myhostname minus the first component.
# $mydomain is used as a default value for many other configuration
# parameters.
#
mydomain = seniord.net

# SENDING MAIL
#
# The myorigin parameter specifies the domain that locally-posted
# mail appears to come from. The default is to append $myhostname,
# which is fine for small sites. If you run a domain with multiple
# machines, you should (1) change this to $mydomain and (2) set up
# a domain-wide alias database that aliases each user to
# user@that.users.mailhost.
#
# For the sake of consistency between sender and recipient addresses,
# myorigin also specifies the default domain name that is appended
# to recipient addresses that have no @domain part.
#
myorigin = $myhostname
#myorigin = $mydomain

# RECEIVING MAIL

# The inet_interfaces parameter specifies the network interface
# addresses that this mail system receives mail on. By default,
# the software claims all active interfaces on the machine. The
# parameter also controls delivery of mail to user@[ip.address].
#
# See also the proxy_interfaces parameter, for network addresses that
# are forwarded to us via a proxy or network address translator.
#
# Note: you need to stop/start Postfix when this parameter changes.
#
inet_interfaces = all
#inet_interfaces = $myhostname
#inet_interfaces = $myhostname, localhost

# The proxy_interfaces parameter specifies the network interface
# addresses that this mail system receives mail on by way of a
```

```

# proxy or network address translation unit. This setting extends
# the address list specified with the inet_interfaces parameter.
#
# You must specify your proxy/NAT addresses when your system is a
# backup MX host for other domains, otherwise mail delivery loops
# will happen when the primary MX host is down.
#
#proxy_interfaces =
#proxy_interfaces = 1.2.3.4

# The mydestination parameter specifies the list of domains that this
# machine considers itself the final destination for.
#
# These domains are routed to the delivery agent specified with the
# local_transport parameter setting. By default, that is the UNIX
# compatible delivery agent that lookups all recipients in /etc/passwd
# and /etc/aliases or their equivalent.
#
# The default is $myhostname + localhost.$mydomain. On a mail domain
# gateway, you should also include $mydomain.
#
# Do not specify the names of virtual domains - those domains are
# specified elsewhere (see sample-virtual.cf).
#
# Do not specify the names of domains that this machine is backup MX
# host for. Specify those names via the relay_domains settings for
# the SMTP server, or use permit_mx_backup if you are lazy (see
# sample-smtpd.cf).
#
# The local machine is always the final destination for mail addressed
# to user@[the.net.work.address] of an interface that the mail system
# receives mail on (see the inet_interfaces parameter).
#
# Specify a list of host or domain names, /file/name or type:table
# patterns, separated by commas and/or whitespace. A /file/name
# pattern is replaced by its contents; a type:table is matched when
# a name matches a lookup key (the right-hand side is ignored).
# Continue long lines by starting the next line with whitespace.
#
# See also below, section "REJECTING MAIL FOR UNKNOWN LOCAL USERS".
#
#mydestination = $myhostname, localhost.$mydomain
mydestination = $myhostname, localhost.$mydomain, $mydomain
#mydestination = $myhostname, localhost.$mydomain, $mydomain,
#    mail.$mydomain, www.$mydomain, ftp.$mydomain

# REJECTING MAIL FOR UNKNOWN LOCAL USERS
#
# The local_recipient_maps parameter specifies optional lookup tables
# with all names or addresses of users that are local with respect
# to $mydestination and $inet_interfaces.
#
# If this parameter is defined, then the SMTP server will reject
# mail for unknown local users. This parameter is defined by default.
#
# To turn off local recipient checking in the SMTP server, specify
# local_recipient_maps = (i.e. empty).

```

```

#
# The default setting assumes that you use the default Postfix local
# delivery agent for local delivery. You need to update the
# local_recipient_maps setting if:
#
# - You define $mydestination domain recipients in files other than
#   /etc/passwd, /etc/aliases, or the $virtual_alias_maps files.
#   For example, you define $mydestination domain recipients in
#   the $virtual_mailbox_maps files.
#
# - You redefine the local delivery agent in master.cf.
#
# - You redefine the "local_transport" setting in main.cf.
#
# - You use the "luser_relay", "mailbox_transport", or "fallback_transport"
#   feature of the Postfix local delivery agent (see sample-local.cf).
#
# Details are described in the LOCAL_RECIPIENT_README file.
#
# Beware: if the Postfix SMTP server runs chrooted, you probably have
# to access the passwd file via the proxymap service, in order to
# overcome chroot restrictions. The alternative, having a copy of
# the system passwd file in the chroot jail is just not practical.
#
# The right-hand side of the lookup tables is conveniently ignored.
# In the left-hand side, specify a bare username, an @domain.tld
# wild-card, or specify a user@domain.tld address.
#
local_recipient_maps = unix:passwd.byname $alias_maps
#local_recipient_maps = proxy:unix:passwd.byname $alias_maps
#local_recipient_maps =

# The unknown_local_recipient_reject_code specifies the SMTP server
# response code when a recipient domain matches $mydestination or
# $inet_interfaces, while $local_recipient_maps is non-empty and the
# recipient address or address local-part is not found.
#
# The default setting is 550 (reject mail) but it is safer to start
# with 450 (try again later) until you are certain that your
# local_recipient_maps settings are OK.
#
#unknown_local_recipient_reject_code = 550
unknown_local_recipient_reject_code = 450

# TRUST AND RELAY CONTROL

# The mynetworks parameter specifies the list of "trusted" SMTP
# clients that have more privileges than "strangers".
#
# In particular, "trusted" SMTP clients are allowed to relay mail
# through Postfix. See the smtpd_recipient_restrictions parameter
# in file sample-smtpd.cf.
#
# You can specify the list of "trusted" network addresses by hand
# or you can let Postfix do it for you (which is the default).
#
# By default (mynetworks_style = subnet), Postfix "trusts" SMTP

```

```

# clients in the same IP subnetworks as the local machine.
# On Linux, this does works correctly only with interfaces specified
# with the "ifconfig" command.
#
# Specify "mynetworks_style = class" when Postfix should "trust" SMTP
# clients in the same IP class A/B/C networks as the local machine.
# Don't do this with a dialup site - it would cause Postfix to "trust"
# your entire provider's network. Instead, specify an explicit
# mynetworks list by hand, as described below.
#
# Specify "mynetworks_style = host" when Postfix should "trust"
# only the local machine.
#
#mynetworks_style = class
#mynetworks_style = subnet
#mynetworks_style = host

# Alternatively, you can specify the mynetworks list by hand, in
# which case Postfix ignores the mynetworks_style setting.
#
# Specify an explicit list of network/netmask patterns, where the
# mask specifies the number of bits in the network part of a host
# address.
#
# You can also specify the absolute pathname of a pattern file instead
# of listing the patterns here. Specify type:table for table-based lookups
# (the value on the table right-hand side is not used).
#
mynetworks = 192.168.0.0/24, 10.0.0.0/24, 127.0.0.0/8
#mynetworks = $config_directory/mynetworks
#mynetworks = hash:/usr/local/etc/postfix/network_table

# The relay_domains parameter restricts what destinations this system will
# relay mail to. See the smtpd_recipient_restrictions restriction in the
# file sample-smtpd.cf for detailed information.
#
# By default, Postfix relays mail
# - from "trusted" clients (IP address matches $mynetworks) to any
destination,
# - from "untrusted" clients to destinations that match $relay_domains or
# subdomains thereof, except addresses with sender-specified routing.
# The default relay_domains value is $mydestination.
#
# In addition to the above, the Postfix SMTP server by default accepts mail
# that Postfix is final destination for:
# - destinations that match $inet_interfaces,
# - destinations that match $mydestination
# - destinations that match $virtual_alias_domains,
# - destinations that match $virtual_mailbox_domains.
# These destinations do not need to be listed in $relay_domains.
#
# Specify a list of hosts or domains, /file/name patterns or type:name
# lookup tables, separated by commas and/or whitespace. Continue
# long lines by starting the next line with whitespace. A file name
# is replaced by its contents; a type:name table is matched when a
# (parent) domain appears as lookup key.
#

```

```
# NOTE: Postfix will not automatically forward mail for domains that
# list this system as their primary or backup MX host. See the
# permit_mx_backup restriction in the file sample-smtpd.cf.
#
relay_domains = $mydestination, $inet_interfaces

# INTERNET OR INTRANET

# The relayhost parameter specifies the default host to send mail to
# when no entry is matched in the optional transport(5) table. When
# no relayhost is given, mail is routed directly to the destination.
#
# On an intranet, specify the organizational domain name. If your
# internal DNS uses no MX records, specify the name of the intranet
# gateway host instead.
#
# In the case of SMTP, specify a domain, host, host:port, [host]:port,
# [address] or [address]:port; the form [host] turns off MX lookups.
#
# If you're connected via UUCP, see also the default_transport parameter.
#
#relayhost = $mydomain
#relayhost = gateway.my.domain
#relayhost = uucphost
relayhost = [192.168.0.5]:25

# REJECTING UNKNOWN RELAY USERS
#
# The relay_recipient_maps parameter specifies optional lookup tables
# with all addresses in the domains that match $relay_domains.
#
# If this parameter is defined, then the SMTP server will reject
# mail for unknown relay users. This feature is off by default.
#
# The right-hand side of the lookup tables is conveniently ignored.
# In the left-hand side, specify an @domain.tld wild-card, or specify
# a user@domain.tld address.
#
#relay_recipient_maps = hash:/usr/local/etc/postfix/relay_recipients

# INPUT RATE CONTROL
#
# The in_flow_delay configuration parameter implements mail input
# flow control. This feature is turned on by default, although it
# still needs further development (it's disabled on SCO UNIX due
# to an SCO bug).
#
# A Postfix process will pause for $in_flow_delay seconds before
# accepting a new message, when the message arrival rate exceeds the
# message delivery rate. With the default 50 SMTP server process
# limit, this limits the mail inflow to 50 messages a second more
# than the number of messages delivered per second.
#
# Specify 0 to disable the feature. Valid delays are 0..10.
#
#in_flow_delay = 1s
```

```
# ADDRESS REWRITING
#
# Insert text from sample-rewrite.cf if you need to do address
# masquerading.
#
# Insert text from sample-canonical.cf if you need to do address
# rewriting, or if you need username->Firstname.Lastname mapping.

# ADDRESS REDIRECTION (VIRTUAL DOMAIN)
#
# Insert text from sample-virtual.cf if you need virtual domain support.

# "USER HAS MOVED" BOUNCE MESSAGES
#
# Insert text from sample-relocated.cf if you need "user has moved"
# style bounce messages. Alternatively, you can bounce recipients
# with an SMTP server access table. See sample-smtpd.cf.

# TRANSPORT MAP
#
# Insert text from sample-transport.cf if you need explicit routing.

# ALIAS DATABASE
#
# The alias_maps parameter specifies the list of alias databases used
# by the local delivery agent. The default list is system dependent.
#
# On systems with NIS, the default is to search the local alias
# database, then the NIS alias database. See aliases(5) for syntax
# details.
#
# If you change the alias database, run "postalias /etc/aliases" (or
# wherever your system stores the mail alias file), or simply run
# "newaliases" to build the necessary DBM or DB file.
#
# It will take a minute or so before changes become visible. Use
# "postfix reload" to eliminate the delay.
#
#alias_maps = dbm:/etc/aliases
#alias_maps = hash:/etc/aliases
#alias_maps = hash:/etc/aliases, nis:mail.aliases
#alias_maps = netinfo:/aliases

# The alias_database parameter specifies the alias database(s) that
# are built with "newaliases" or "sendmail -bi". This is a separate
# configuration parameter, because alias_maps (see above) may specify
# tables that are not necessarily all under control by Postfix.
#
#alias_database = dbm:/etc/aliases
#alias_database = dbm:/etc/mail/aliases
#alias_database = hash:/etc/aliases
#alias_database = hash:/etc/aliases, hash:/opt/majordomo/aliases

# ADDRESS EXTENSIONS (e.g., user+foo)
#
# The recipient_delimiter parameter specifies the separator between
# user names and address extensions (user+foo). See canonical(5),
```

```

# local(8), relocated(5) and virtual(5) for the effects this has on
# aliases, canonical, virtual, relocated and .forward file lookups.
# Basically, the software tries user+foo and .forward+foo before
# trying user and .forward.
#
#recipient_delimiter = +

# DELIVERY TO MAILBOX
#
# The home_mailbox parameter specifies the optional pathname of a
# mailbox file relative to a user's home directory. The default
# mailbox file is /var/spool/mail/user or /var/mail/user. Specify
# "Maildir/" for qmail-style delivery (the / is required).
#
home_mailbox = Mailbox
#home_mailbox = Maildir/

# The mail_spool_directory parameter specifies the directory where
# UNIX-style mailboxes are kept. The default setting depends on the
# system type.
#
#mail_spool_directory = /var/mail
#mail_spool_directory = /var/spool/mail

# The mailbox_command parameter specifies the optional external
# command to use instead of mailbox delivery. The command is run as
# the recipient with proper HOME, SHELL and LOGNAME environment settings.
# Exception: delivery for root is done as $default_user.
#
# Other environment variables of interest: USER (recipient username),
# EXTENSION (address extension), DOMAIN (domain part of address),
# and LOCAL (the address localpart).
#
# Unlike other Postfix configuration parameters, the mailbox_command
# parameter is not subjected to $parameter substitutions. This is to
# make it easier to specify shell syntax (see example below).
#
# Avoid shell meta characters because they will force Postfix to run
# an expensive shell process. Procmail alone is expensive enough.
#
# IF YOU USE THIS TO DELIVER MAIL SYSTEM-WIDE, YOU MUST SET UP AN
# ALIAS THAT FORWARDS MAIL FOR ROOT TO A REAL USER.
#
#mailbox_command = /some/where/procmail
#mailbox_command = /some/where/procmail -a "$EXTENSION"

# The mailbox_transport specifies the optional transport in master.cf
# to use after processing aliases and .forward files. This parameter
# has precedence over the mailbox_command, fallback_transport and
# luser_relay parameters.
#
# Specify a string of the form transport:nexthop, where transport is
# the name of a mail delivery transport defined in master.cf. The
# :nexthop part is optional. For more details see the sample transport
# configuration file.
#
# NOTE: if you use this feature for accounts not in the UNIX password

```

```

# file, then you must update the "local_recipient_maps" setting in
# the main.cf file, otherwise the SMTP server will reject mail for
# non-UNIX accounts with "User unknown in local recipient table".
#
#mailbox_transport = lmtp:unix:/file/name
#mailbox_transport = cyrus

# The fallback_transport specifies the optional transport in master.cf
# to use for recipients that are not found in the UNIX passwd database.
# This parameter has precedence over the luser_relay parameter.
#
# Specify a string of the form transport:nexthop, where transport is
# the name of a mail delivery transport defined in master.cf. The
# :nexthop part is optional. For more details see the sample transport
# configuration file.
#
# NOTE: if you use this feature for accounts not in the UNIX password
# file, then you must update the "local_recipient_maps" setting in
# the main.cf file, otherwise the SMTP server will reject mail for
# non-UNIX accounts with "User unknown in local recipient table".
#
#fallback_transport = lmtp:unix:/file/name
#fallback_transport = cyrus
#fallback_transport =

# The luser_relay parameter specifies an optional destination address
# for unknown recipients. By default, mail for unknown@$mydestination
# and unknown@[inet_interfaces] is returned as undeliverable.
#
# The following expansions are done on luser_relay: $user (recipient
# username), $shell (recipient shell), $home (recipient home directory),
# $recipient (full recipient address), $extension (recipient address
# extension), $domain (recipient domain), $local (entire recipient
# localpart), $recipient_delimiter. Specify ${name?value} or
# ${name:value} to expand value only when $name does (does not) exist.
#
# luser_relay works only for the default Postfix local delivery agent.
#
# NOTE: if you use this feature for accounts not in the UNIX password
# file, then you must specify "local_recipient_maps =" (i.e. empty) in
# the main.cf file, otherwise the SMTP server will reject mail for
# non-UNIX accounts with "User unknown in local recipient table".
#
#luser_relay = $user@other.host
#luser_relay = $local@other.host
#luser_relay = admin+$local

# JUNK MAIL CONTROLS
#
# The controls listed here are only a very small subset. See the file
# sample-smtpd.cf for an elaborate list of anti-UCE controls.

# The header_checks parameter specifies an optional table with patterns
# that each logical message header is matched against, including
# headers that span multiple physical lines.
#
# By default, these patterns also apply to MIME headers and to the

```

```
# headers of attached messages. With older Postfix versions, MIME and
# attached message headers were treated as body text.
#
# For details, see the sample-filter.cf file.
#
#header_checks = regexp:/usr/local/etc/postfix/header_checks

# FAST ETRN SERVICE
#
# Postfix maintains per-destination logfiles with information about
# deferred mail, so that mail can be flushed quickly with the SMTP
# "ETRN domain.tld" command, or by executing "sendmail -qRdomain.tld".
#
# By default, Postfix maintains deferred mail logfile information
# only for destinations that Postfix is willing to relay to (as
# specified in the relay_domains parameter). For other destinations,
# Postfix attempts to deliver ALL queued mail after receiving the
# SMTP "ETRN domain.tld" command, or after execution of "sendmail
# -qRdomain.tld". This can be slow when a lot of mail is queued.
#
# The fast_flush_domains parameter controls what destinations are
# eligible for this "fast ETRN/sendmail -qR" service.
#
#fast_flush_domains = $relay_domains
#fast_flush_domains =

# SHOW SOFTWARE VERSION OR NOT
#
# The smtpd_banner parameter specifies the text that follows the 220
# code in the SMTP server's greeting banner. Some people like to see
# the mail version advertised. By default, Postfix shows no version.
#
# You MUST specify $myhostname at the start of the text. That is an
# RFC requirement. Postfix itself does not care.
#
#smtpd_banner = $myhostname ESMTP $mail_name
#smtpd_banner = $myhostname ESMTP $mail_name ($mail_version)

# PARALLEL DELIVERY TO THE SAME DESTINATION
#
# How many parallel deliveries to the same user or domain? With local
# delivery, it does not make sense to do massively parallel delivery
# to the same user, because mailbox updates must happen sequentially,
# and expensive pipelines in .forward files can cause disasters when
# too many are run at the same time. With SMTP deliveries, 10
# simultaneous connections to the same domain could be sufficient to
# raise eyebrows.
#
# Each message delivery transport has its XXX_destination_concurrency_limit
# parameter. The default is $default_destination_concurrency_limit for
# most delivery transports. For the local delivery agent the default is 2.

local_destination_concurrency_limit = 5
#default_destination_concurrency_limit = 10

# DEBUGGING CONTROL
#
```

```

# The debug_peer_level parameter specifies the increment in verbose
# logging level when an SMTP client or server host name or address
# matches a pattern in the debug_peer_list parameter.
#
debug_peer_level = 2

# The debug_peer_list parameter specifies an optional list of domain
# or network patterns, /file/name patterns or type:name tables. When
# an SMTP client or server host name or address matches a pattern,
# increase the verbose logging level by the amount specified in the
# debug_peer_level parameter.
#
#debug_peer_list = 127.0.0.1
#debug_peer_list = some.domain

# The debugger_command specifies the external command that is executed
# when a Postfix daemon program is run with the -D option.
#
# Use "command .. & sleep 5" so that the debugger can attach before
# the process marches on. If you use an X-based debugger, be sure to
# set up your XAUTHORITY environment variable before starting Postfix.
#
debugger_command =
    PATH=/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin
    xgdb $daemon_directory/$process_name $process_id & sleep 5

# If you don't have X installed on the Postfix machine, try:
# debugger_command =
#     PATH=/bin:/usr/bin:/usr/local/bin; export PATH; (echo cont;
#     echo where) | gdb $daemon_directory/$process_name $process_id 2>&1
#     >$config_directory/$process_name.$process_id.log & sleep 5

# INSTALL-TIME CONFIGURATION INFORMATION
#
# The following parameters are used when installing a new Postfix version.
#
# sendmail_path: The full pathname of the Postfix sendmail command.
# This is the Sendmail-compatible mail posting interface.
#
sendmail_path = /usr/local/sbin/sendmail

# newaliases_path: The full pathname of the Postfix newaliases command.
# This is the Sendmail-compatible command to build alias databases.
#
newaliases_path = /usr/local/bin/newaliases

# mailq_path: The full pathname of the Postfix mailq command. This
# is the Sendmail-compatible mail queue listing command.
#
mailq_path = /usr/local/bin/mailq

# setgid_group: The group for mail submission and queue management
# commands. This must be a group name with a numerical group ID that
# is not shared with other accounts, not even with the Postfix account.
#
setgid_group = maildrop

```

```
# manpage_directory: The location of the Postfix on-line manual pages.
#
manpage_directory = /usr/local/man

# sample_directory: The location of the Postfix sample configuration files.
#
sample_directory = /usr/local/etc/postfix

# readme_directory: The location of the Postfix README files.
#
readme_directory = no

# other options
default_process_limit = 200
```

## Appendix G – Performance Index

Computer Tested	BYTE UNIX Benchmark						NetPerf	
	FreeBSD Version	Arithmetic Test	Dhrystone Test	Exec Test	Context Switching	Shell Scripts	Average	Network Throughput (loopback interface)
SERVER1 - Pentium II 350Mhz, 256 MB SD-RAM, 9 Gigabyte 10K RPM SCSI	4.7-RELEASE	36.4	31	23.3	48.7	31	34.08	428.12 Megabits
AMD Duron 950Mhz, 192MB SD-RAM, 40GB 7200RPM IDE.	4.4-RELEASE	153.9	83.5	29.9	118.8	41.9	85.6	375.73 Megabits
Pentium III 800Mhz, 256 MB SD-RAM, 20 Gigabyte 7200 RPM IDE	4.4-RELEASE	79.9	72.2	29.7	108.7	22.3	62.56	1205.68 Megabits
Celeron 1Ghz, 256 MB SD-RAM, 20 Gigabyte 7200RPM IDE	4.4-RELEASE	99.3	89.9	57.9	141.3	73.2	92.32	1548.54 Megabits
Pentium III 700Mhz, 384 MB SD-RAM, 10 Gigabyte 5400 RPM IDE, LAPTOP	5.0-RELEASE	86	76.8	28.8	48.9	42.2	56.54	979.45 Megabits
Celeron 533Mhz, 64MB SD-RAM, 6.4GB 7200RPM IDE	4.7-RELEASE	55.4	46.5	22.7	73.8	35.2	46.72	266.03 Megabits

